下载APP

08 | 哨兵集群：哨兵挂了，主从库还能切换吗？

2020-08-21 蒋德钧

Redis核心技术与实战

进入课程 >

讲述：蒋德钧
时长 14:30　大小 13.28M

你好，我是蒋德钧。

上节课，我们学习了哨兵机制，它可以实现主从库的自动切换。通过部署多个实例，就形成了一个哨兵集群。哨兵集群中的多个实例共同判断，可以降低对主库下线的误判率。

但是，我们还是要考虑一个问题：如果有哨兵实例在运行时发生了故障，主从库还能正常切换吗？

实际上，一旦多个实例组成了**哨兵集群**，即使有哨兵实例出现故障挂掉了，其他哨兵还能继续协作完成主从库切换的工作，包括判定主库是不是处于下线状态，选择新主库，以及通知从库和客户端。

如果你部署过哨兵集群的话就会知道，在配置哨兵的信息时，我们只需要用到下面的这个配置项，设置**主库的 IP** 和**端口**，并没有配置其他哨兵的连接信息。

```
1 sentinel monitor <master-name> <ip> <redis-port> <quorum>
```

复制代码

这些哨兵实例既然都不知道彼此的地址，又是怎么组成集群的呢？要弄明白这个问题，我们就需要学习一下哨兵集群的组成和运行机制了。

基于 pub/sub 机制的哨兵集群组成

哨兵实例之间可以相互发现，要归功于 Redis 提供的 pub/sub 机制，也就是发布 / 订阅机制。

哨兵只要和主库建立起了连接，就可以在主库上发布消息了，比如说发布它自己的连接信息（IP 和端口）。同时，它也可以从主库上订阅消息，获得其他哨兵发布的连接信息。当多个哨兵实例都在主库上做了发布和订阅操作后，它们之间就能知道彼此的 IP 地址和端口。

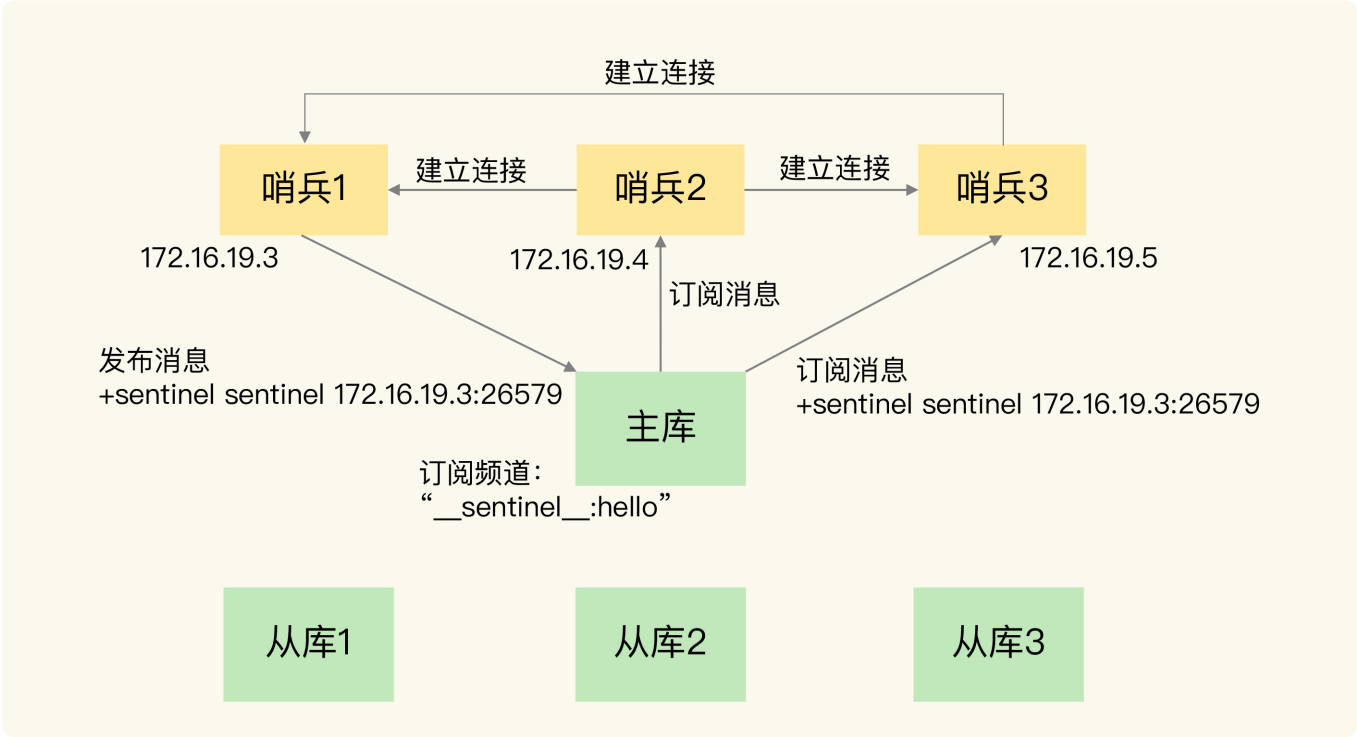
除了哨兵实例，我们自己编写的应用程序也可以通过 Redis 进行消息的发布和订阅。所以，为了区分不同应用的消息，Redis 会以频道的形式，对这些消息进行分门别类的管理。所谓的频道，实际上就是消息的类别。当消息类别相同时，它们就属于同一个频道。反之，就属于不同的频道。**只有订阅了同一个频道的应用，才能通过发布的消息进行信息交换。**

在主从集群中，主库上有一个名为“__sentinel__:hello”的频道，不同哨兵就是通过它来相互发现，实现互相通信的。

我来举个例子，具体说明一下。在下图中，哨兵 1 把自己的 IP（172.16.19.3）和端口（26579）发布到“__sentinel__:hello”频道上，哨兵 2 和 3 订阅了该频道。那么此时，哨兵 2 和 3 就可以从这个频道直接获取哨兵 1 的 IP 地址和端口号。

然后，哨兵 2、3 可以和哨兵 1 建立网络连接。通过这个方式，哨兵 2 和 3 也可以建立网络连接，这样一来，哨兵集群就形成了。它们相互间可以通过网络连接进行通信，比如说

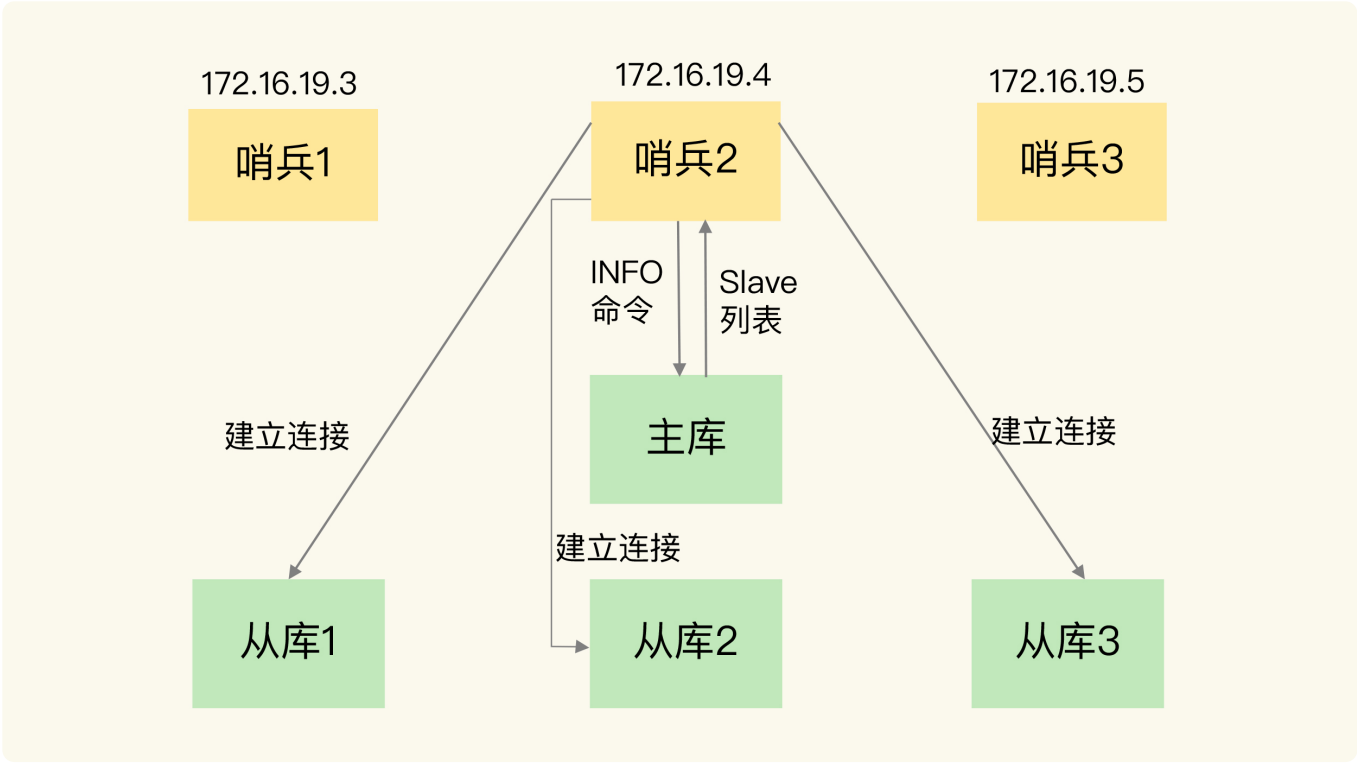
对主库有没有下线这件事儿进行判断和协商。



哨兵除了彼此之间建立起连接形成集群外，还需要和从库建立连接。这是因为，在哨兵的监控任务中，它需要对主从库都进行心跳判断，而且在主从库切换完成后，它还需要通知从库，让它们和新主库进行同步。

那么，**哨兵是如何知道从库的 IP 地址和端口的呢？**

这是由哨兵向主库发送 INFO 命令来完成的。就像下图所示，哨兵 2 给主库发送 INFO 命令，主库接受到这个命令后，就会把从库列表返回给哨兵。接着，哨兵就可以根据从库列表中的连接信息，和每个从库建立连接，并在这个连接上持续地对从库进行监控。哨兵 1 和 3 可以通过相同的方法和从库建立连接。



你看，通过 pub/sub 机制，哨兵之间可以组成集群，同时，哨兵又通过 INFO 命令，获得了从库连接信息，也能和从库建立连接，并进行监控了。

但是，哨兵不能只和主、从库连接。因为，主从库切换后，客户端也需要知道新主库的连接信息，才能向新主库发送请求操作。所以，哨兵还需要完成把新主库的信息告诉客户端这个任务。

而且，在实际使用哨兵时，我们有时会遇到这样的问题：如何在客户端通过监控了解哨兵进行主从切换的过程呢？比如说，主从切换进行到哪一步了？这其实就是要求，客户端能够获取到哨兵集群在监控、选主、切换这个过程中发生的各种事件。

此时，我们仍然可以依赖 pub/sub 机制，来帮助我们完成哨兵和客户端间的信息同步。

基于 pub/sub 机制的客户端事件通知

从本质上说，哨兵就是一个运行在特定模式下的 Redis 实例，只不过它并不服务请求操作，只是完成监控、选主和通知的任务。所以，每个哨兵实例也提供 pub/sub 机制，客户端可以从哨兵订阅消息。哨兵提供的消息订阅频道有很多，不同频道包含了主从库切换过程中的不同关键事件。

频道有这么多，一下子全部学习容易丢失重点。为了减轻你的学习压力，我把重要的频道汇总在了一起，涉及几个关键事件，包括主库下线判断、新主库选定、从库重新配置。

事件	相关频道
主库下线事件	+sdown（实例进入“主观下线”状态）
	-sdown（实例退出“主观下线”状态）
	+odown（实例进入“客观下线”状态）
	-odown（实例退出“客观下线”状态）
从库重新配置事件	+slave-reconf-sent （哨兵发送SLAVEOF命令重新配置从库）
	+slave-reconf-inprog （从库配置了新主库，但尚未进行同步）
	+slave-reconf-done（从库配置了新主库，且和新主库完成同步）
新主库切换	+switch-master（主库地址发生变化）

知道了这些频道之后，你就可以**让客户端从哨兵这里订阅消息**了。具体的操作步骤是，客户端读取哨兵的配置文件后，可以获得哨兵的地址和端口，和哨兵建立网络连接。然后，我们可以在客户端执行订阅命令，来获取不同的事件消息。

举个例子，你可以执行如下命令，来订阅“所有实例进入客观下线状态的事件”：

复制代码

```
1 SUBSCRIBE +odown
```

当然，你也可以执行如下命令，订阅所有的事件：

复制代码

```
1 PSUBSCRIBE *
```

当哨兵把新主库选择出来后，客户端就会看到下面的 switch-master 事件。这个事件表示主库已经切换了，新主库的 IP 地址和端口信息已经有了。这个时候，客户端就可以用这里

面的新主库地址和端口进行通信了。

```
switch-master <master name> <oldip> <oldport> <newip> <newport>
```

有了这些事件通知，客户端不仅可以在主从切换后得到新主库的连接信息，还可以监控到主从库切换过程中发生的各个重要事件。这样，客户端就可以知道主从切换进行到哪一步了，有助于了解切换进度。

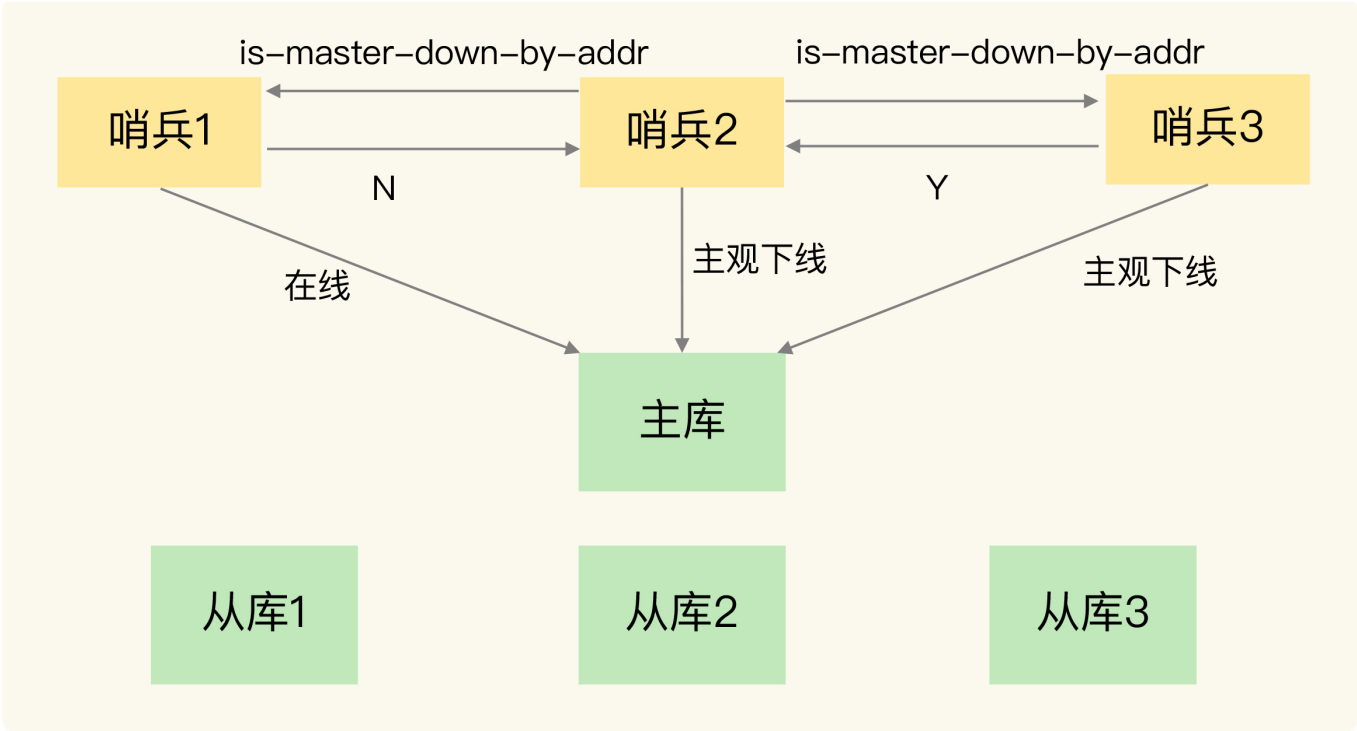
好了，有了 pub/sub 机制，哨兵和哨兵之间、哨兵和从库之间、哨兵和客户端之间就都能建立起连接了，再加上我们上节课介绍主库下线判断和选主依据，哨兵集群的监控、选主和通知三个任务就基本可以正常工作了。不过，我们还需要考虑一个问题：主库故障以后，哨兵集群有多个实例，那怎么确定由哪个哨兵来进行实际的主从切换呢？

由哪个哨兵执行主从切换？

确定由哪个哨兵执行主从切换的过程，和主库“客观下线”的判断过程类似，也是一个“投票仲裁”的过程。在具体了解这个过程前，我们再来看下，判断“客观下线”的仲裁过程。

哨兵集群要判定主库“客观下线”，需要有一定数量的实例都认为该主库已经“主观下线”了。我在上节课向你介绍了判断“客观下线”的原则，接下来，我介绍下具体的判断过程。

任何一个实例只要自身判断主库“主观下线”后，就会给其他实例发送 is-master-down-by-addr 命令。接着，其他实例会根据自己和主库的连接情况，做出 Y 或 N 的响应，Y 相当于赞成票，N 相当于反对票。



一个哨兵获得了仲裁所需的赞成票数后，就可以标记主库为“客观下线”。这个所需的赞成票数是通过哨兵配置文件中的 quorum 配置项设定的。例如，现在有 5 个哨兵，quorum 配置的是 3，那么，一个哨兵需要 3 张赞成票，就可以标记主库为“客观下线”了。这 3 张赞成票包括哨兵自己的一张赞成票和另外两个哨兵的赞成票。

此时，这个哨兵就可以再给其他哨兵发送命令，表明希望由自己来执行主从切换，并让所有其他哨兵进行投票。这个投票过程称为“Leader 选举”。因为最终执行主从切换的哨兵称为 Leader，投票过程就是确定 Leader。

在投票过程中，任何一个想成为 Leader 的哨兵，要满足两个条件：第一，拿到半数以上的赞成票；第二，拿到的票数同时还需要大于等于哨兵配置文件中的 quorum 值。以 3 个哨兵为例，假设此时的 quorum 设置为 2，那么，任何一个想成为 Leader 的哨兵只要拿到 2 张赞成票，就可以了。

这么说你可能还不太好理解，我再画一张图片，展示一下 3 个哨兵、quorum 为 2 的选举过程。

时间	哨兵1（S1）	哨兵2（S2）	哨兵3（S3）
T1	给自己投1票Y 向S2、S3发投票请求， 表示要成为Leader		
T2			给自己投1票Y 向S1、S2发投票请求， 表示要成为Leader
T3	收到S3的请求，回复N	收到S3的请求，回复Y	
T4		收到S1的请求，回复N	
T5	1票Y，1票N		2票Y，成为Leader

在 T1 时刻，S1 判断主库为“客观下线”，它想成为 Leader，就先给自己投一张赞成票，然后分别向 S2 和 S3 发送命令，表示要成为 Leader。

在 T2 时刻，S3 判断主库为“客观下线”，它也想成为 Leader，所以也先给自己投一张赞成票，再分别向 S1 和 S2 发送命令，表示要成为 Leader。

在 T3 时刻，S1 收到了 S3 的 Leader 投票请求。因为 S1 已经给自己投了一票 Y，所以它不能再给其他哨兵投赞成票了，所以 S1 回复 N 表示不同意。同时，S2 收到了 T2 时 S3 发送的 Leader 投票请求。因为 S2 之前没有投过票，它会给第一个向它发送投票请求的哨兵回复 Y，给后续再发送投票请求的哨兵回复 N，所以，在 T3 时，S2 回复 S3，同意 S3 成为 Leader。

在 T4 时刻，S2 才收到 T1 时 S1 发送的投票命令。因为 S2 已经在 T3 时同意了 S3 的投票请求，此时，S2 给 S1 回复 N，表示不同意 S1 成为 Leader。发生这种情况，是因为 S3 和 S2 之间的网络传输正常，而 S1 和 S2 之间的网络传输可能正好拥塞了，导致投票请求传输慢了。

最后，在 T5 时刻，S1 得到的票数是来自它自己的一票 Y 和来自 S2 的一票 N。而 S3 除了自己的赞成票 Y 以外，还收到了来自 S2 的一票 Y。此时，S3 不仅获得了半数以上的 Leader 赞成票，也达到预设的 quorum 值（quorum 为 2），所以它最终成为了 Leader。接着，S3 会开始执行选主操作，而且在选定新主库后，会给其他从库和客户端通知新主库的信息。

如果 S3 没有拿到 2 票 Y，那么这轮投票就不会产生 Leader。哨兵集群会等待一段时间（也就是哨兵故障转移超时时间的 2 倍），再重新选举。这是因为，哨兵集群能够进行成功投票，很大程度上依赖于选举命令的正常网络传播。如果网络压力较大或有短时堵塞，就可能导致没有一个哨兵能拿到半数以上的赞成票。所以，等到网络拥塞好转之后，再进行投票选举，成功的概率就会增加。

需要注意的是，如果哨兵集群只有 2 个实例，此时，一个哨兵要想成为 Leader，必须获得 2 票，而不是 1 票。所以，如果有个哨兵挂掉了，那么，此时的集群是无法进行主从库切换的。因此，通常我们至少会配置 3 个哨兵实例。这一点很重要，你在实际应用时可不能忽略了。

小结

通常，我们在解决一个系统问题的时候，会引入一个新机制，或者设计一层新功能，就像我们在这两节课学习的内容：为了实现主从切换，我们引入了哨兵；为了避免单个哨兵故障后无法进行主从切换，以及为了减少误判率，又引入了哨兵集群；哨兵集群又需要有一些机制来支撑它的正常运行。

这节课上，我就向你介绍了支持哨兵集群的这些关键机制，包括：

基于 pub/sub 机制的哨兵集群组成过程；

基于 INFO 命令的从库列表，这可以帮助哨兵和从库建立连接；

基于哨兵自身的 pub/sub 功能，这实现了客户端和哨兵之间的事件通知。

对于主从切换，当然不是哪个哨兵想执行就可以执行的，否则就乱套了。所以，这就需要哨兵集群在判断了主库“客观下线”后，经过投票仲裁，选举一个 Leader 出来，由它负责实际的主从切换，即由它来完成新主库的选择以及通知从库与客户端。

最后，我想再给你分享一个经验：**要保证所有哨兵实例的配置是一致的，尤其是主观下线的判断值 down-after-milliseconds**。我们曾经就踩过一个“坑”。当时，在我们的项目中，因为这个值在不同的哨兵实例上配置不一致，导致哨兵集群一直没有对有故障的主库形成共识，也就没有及时切换主库，最终的结果就是集群服务不稳定。所以，你一定不要忽略这条看似简单的经验。

每课一问

这节课上，我给你提一个小问题。

假设有一个 Redis 集群，是“一主四从”，同时配置了包含 5 个哨兵实例的集群，quorum 值设为 2。在运行过程中，如果有 3 个哨兵实例都发生故障了，此时，Redis 主库如果有故障，还能正确地判断主库“客观下线”吗？如果可以的话，还能进行主从库自动切换吗？此外，哨兵实例是不是越多越好呢，如果同时调大 down-after-milliseconds 值，对减少误判是不是也有好处呢？

欢迎你在留言区跟我交流讨论。如果你身边也有要学习哨兵集群相关知识点的朋友，也欢迎你能帮我把今天的内容分享给他们，帮助他们一起解决问题。我们下节课见。

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 07 | 哨兵机制：主库挂了，如何不间断服务？

下一篇 09 | 切片集群：数据增多了，是该加内存还是加实例？

精选留言 (20)

写留言

小喵喵 置顶

2020-08-21

老师请教下：
1、图示哨兵选举过程中，选举的结果取决于S2的投票，如果S2也投给自己，并且每轮投票都是只投给自己，岂不是无法选出“Leader”，是不是这个过程从了死循环呢？
2、投票投给谁，依据是什么？

展开

作者回复: 文章读得很仔细！

先回答第一个问题：
文章中的例子里，要发生S1、S2和S3同时同自己投票的情况，这需要这三个哨兵基本同时判定了主库客观下线。但是，不同哨兵的网络连接、系统压力不完全一样，接收到下线协商消息的时间

也可能不同，所以，它们同时做出主库客观下线判定的概率较小，一般都有个先后关系。文章中的例子，就是S1、S3先判定，S2一直没有判定。

其次，哨兵对主从库进行的在线状态检查等操作，是属于一种时间事件，用一个定时器来完成，一般来说每100ms执行一次这些事件。每个哨兵的定时器执行周期都会加上一个小小的随机时间偏移，目的是让每个哨兵执行上述操作的时间能稍微错开些，也是为了避免它们都同时判定主库下线，同时选举Leader。

最后，即使出现了都投给自己一票的情况，导致无法选出Leader，哨兵会停一段时间（一般是故障转移超时时间failover_timeout的2倍），然后再可以进行下一轮投票。

第二个问题：哨兵如果没有给自己投票，就会把票投给第一个给它发送投票请求的哨兵。后续再有投票请求来，哨兵就拒接投票了。

2

Kaito
2020-08-21

Redis 1主4从，5个哨兵，哨兵配置quorum为2，如果3个哨兵故障，当主库宕机时，哨兵能否判断主库“客观下线”？能否自动切换?

经过实际测试，我的结论如下：
...
展开

作者回复: Kaito同学的答题一如既往的精彩!

24 56

Darren
2020-08-21

1、可以正确的判断主库“客观下线”，以为其中一个哨兵已经获得了“客观下线”所需要的投票数；
2、不能进行自动的主从切换，因为在主从切换的时候，必须选择出一个主哨兵，但是选择主哨兵有2个条件：
2.1 拿到半数以上的赞成票；...
展开

2 4

kingdompeak

2020-08-21

1.可以判断主库“客观下线”。比如哨兵实例1判断主库为客观下线，然后向哨兵实例2发送is-master-down-by-addr命令，如果哨兵实例2此时也判断主库为客观下线，就会返回Y，此时哨兵实例1就会有两个Y（包括自己的），满足配置项quorum，所以就可以判断主库客观下线。
2.不能进行主从进行主从切换前需要选执行切换操作的Leader。由于两个哨兵实例在选L...
展开

2 3

riryoutexi
2020-08-21

整个哨兵集群都挂了，还会主从切换吗
展开

作者回复: 哨兵都挂了，无能为力了。。。

1 1

Q
2020-08-21

干货满满。。 最近一直在测试哨兵集群，get 到一个点: 自己还要给自己投一票！也就是每个哨兵只有一次投票权，投自己或别人！
展开

1 1

test
2020-08-21

可以判断客观下线，但是无法进行选主。调大参数对误判有好处。

1

一步
2020-08-24

在哨兵节点选取Leader 节点的时候，某个节点已经投出了 Y 票，但是该轮没有选举出来Leader, 这时候这个节点怎么知道已经到了下一轮需要继续投票了呢？ 也可以这样问：一个节点在一轮只能投出一个票，但是这个节点怎么知道一轮什么时候，什么时候结束的呢？
展开

吕
2020-08-24

这篇文章太好了，直接解决了一个困惑我很久的问题，我一直把判断下线和选主主当成了同一件事，把quorum当成是判断下线和选举leader的阀值，原来判断下线还选主是两个分开的事情

作者回复: 有帮助就好 :D

heyman
2020-08-24

老师，请教一下：哨兵和主库建立连接后，通过info来获取从库信息。那么，后续从库有新增或摘除，哨兵是怎样知道的呢？

展开

注定非凡
2020-08-23

一，作者讲了什么？
哨兵集群的工作机制

二，作者是怎么把这事给讲明白的？
1，哨兵之间互通机制：基于pub/sub机制，在主库中有一个"__sentinel__:hello"的频...

展开

Kyushu
2020-08-23

应该是类似于Raft的Leader选举，但是我记得哨兵这里应该还有一个majority的配置吧，不是超过半数再执行故障转移吧？

展开

yyl
2020-08-22

解答：
1. 主观下线 票数达到quorum即可标记客观下线。因此仍可正确判断主库“客观下线”。
2. 哨兵Leader选举 票数需达到半数，且大于等于quorum。Leader选举可正确进行，因此仍可完成主从自动切换。
3. 哨兵实例，并非越多越好。其一，哨兵需要与主从库和客户端建立连接，实例越多，...

展开

yyl
2020-08-22

“客户端读取哨兵的配置文件后，可以获得哨兵的地址和端口，和哨兵建立网络连接。然后，我们可以在客户端执行订阅命令，来获取不同的事件消息。”

客户端需要与所有的哨兵实例建立网络连接吗？

展开

范闲
2020-08-22

哨兵判断下线分为可能下线和确定下线两种状态。
在课后的例子中，5个哨兵正常2个，异常3个，qurum为2（判断确定下线的哨兵数目）

根据主从选举要求必须半数以上的节点同意，即要求数量大于N/2+1。此例中是5/2+1 = 3，而只有2个哨兵活着因此不可能完成主从切换。...

展开

作者回复: 理解到位了！

不过，一般我们还是叫主观下线和客观下线更多些。。

盟讯
2020-08-21

可以判断客观下线，两个哨兵都会判断“主观下线”，达到仲裁值所需要的数：2。
不会进行主从切换，因为在哨兵选择leader时，每一个哨兵都会选择自己，票数问题相等

哨兵实例不是越多越好，实例越多通信越频繁，会造成网络拥塞。...

展开

倪大人
2020-08-21

问个问题，文章里的S1、S2、S3那张图，为什么S2不能给自己投票？是不是在“客观下

线”中投了赞成票的哨兵才能竞选leader?

展开

zhou

2020-08-21

因为只有两个实例，quorum 是 2，所以两个实例必须都判断为主观下线，才会确认为客观下线。

但只要其中一个实例确认为客观下线，另一个实例必然也会确认为客观下线。此时两个实例都希望申请成为 Leader，先给自己投票，然后请求对方投票。由于都已给自己投过票...

展开

Kirito

2020-08-21

会有3个哨兵都投给自己的情况吗？那不是平票了😂

展开

小贤

2020-08-21

down-after-milliseconds 这个参数一般设置多大合适呢?